Joint Center for Satellite Data Assimilation

Office Note (unassigned)

CRTM: Cloud and Aerosol Optical Property Lookup Table Interpolation Speedup for REL-1.2

Paul van Delst[a]
JCSDA/EMC/SAIC

December, 2008

---

[a]paul.vandelst@noaa.gov

Change History

Date	Author	Change
2008-12-04	P.van Delst	Initial release.
2008-12-31	P.van Delst	Consolidated the CloudScatter and AerosolScatter timing results. Added the CRTM ComponentTest linux/gfortran timing results

1 Introduction

A central design principle of the components of the CRTM Tangent-linear (TL) and Adjoint (AD) models is that the Forward model is always called first. Thus, in principle, forward model calculations are always available for the various TL and AD components.

The REL-1.1 CRTM AtmScatter modules obtained the interpolated cloud and aerosol optical properties from the CloudCoeff and AerosolCoeff lookup tables (LUTs) as shown in figure 1.1. For the tangent-linear and adjoint models (figures 1.1(b) and (c)), the necessary interpolation parameters such as the bracketing indices and interpolating polynomials were always recomputed.

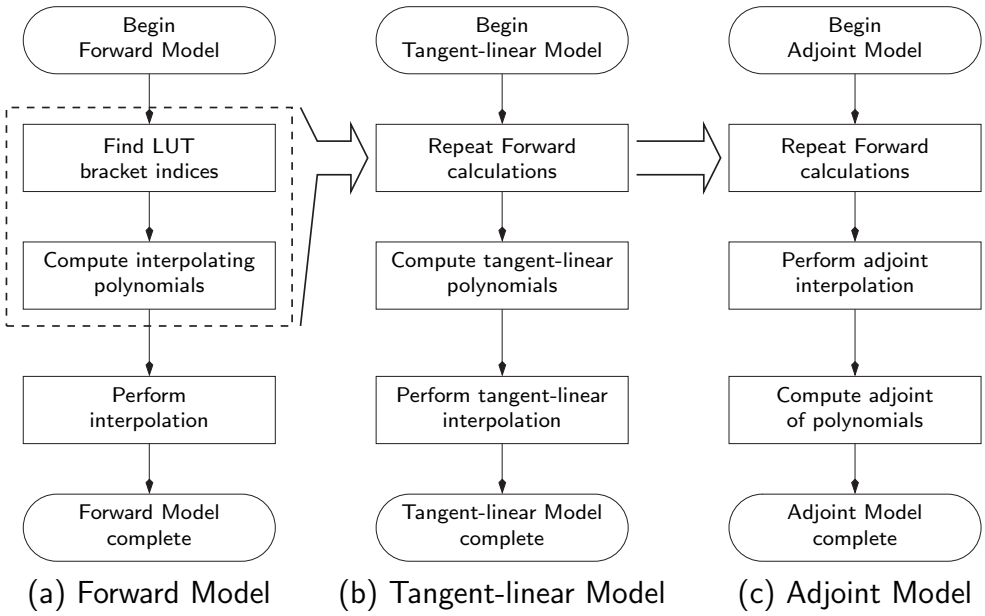

Figure 1.1: Schematic flowcharts of the REL-1.1 CRTM Forward, Tangent-linear, and Adjoint routines used to interpolate the cloud and aerosol optical properties from the CloudCoeff and AerosolCoeff lookup tables. The forward component of the interpolation is recomputed in the tangent-linear and adjoint routines.

The CRTM already saves the intermediate forward model results for its components in private structures, i.e. the structure definition is public, but the internals are private. We call these structures the internal variables for the particular CRTM component (e.g. AtmAbsorption, CloudScatter, AerosolScatter, etc).

To avoid the unnecessary forward model recalculations for the cloud and aerosol property LUT interpolations, separate structures were created to retain the intermediate LUT interpolation results, and added to the main internal variable structure. The structure used for cloudy atmospheres is shown in figure 1.2 and its usage in the CloudScatter internal variable structure is shown in figure 1.3. Similar, the aerosol structure is shown in figure 1.4, and its usage in the AerosolScatter internal variable structure is shown in figure 1.5. Because the cloud optical properties in the microwave are dependent upon an additional parameter (temperature), a different structure is defined for cloud and aerosol computations to minimise memory usage.

With the intermediate interpolation results now available for reuse, the tangent-linear and adjoint routines now avoid the forward model recalculations altogether, as shown schematically in figure 1.6.

The timing data that follows compares the CloudScatter and AerosolScatter component tests for a baseline (revision 2757) and modified (revision 2787) version of the CRTM library from the RB-1.2 branch.

```
TYPE :: CSinterp_type
  ! The interpolating polynomials
  TYPE(LPoly_type) :: wlp  ! Frequency
  TYPE(LPoly_type) :: xlp  ! Effective radius
  TYPE(LPoly_type) :: ylp  ! Temperature
  ! The LUT interpolation indices
  INTEGER :: i1, i2        ! Frequency
  INTEGER :: j1, j2        ! Effective radius
  INTEGER :: k1, k2        ! Temperature
  ! The LUT interpolation boundary check
  LOGICAL :: f_outbound    ! Frequency
  LOGICAL :: r_outbound    ! Effective radius
  LOGICAL :: t_outbound    ! Temperature
  ! The interpolation input
  REAL(fp) :: f_int        ! Frequency
  REAL(fp) :: r_int        ! Effective radius
  REAL(fp) :: t_int        ! Temperature
  ! The data to be interpolated
  REAL(fp) :: f(NPTS)      ! Frequency
  REAL(fp) :: r(NPTS)      ! Effective radius
  REAL(fp) :: t(NPTS)      ! Temperature
END TYPE CSinterp_type
```

Figure 1.2: The structure definition to hold the forward model cloud optical properties lookup table interpolation results.

```
TYPE :: CRTM_CSVariables_type
  PRIVATE
  ! The interpolation data
  TYPE(CSinterp_type) :: csi(MAX_N_LAYERS, MAX_N_CLOUDS)
  ! The interpolation results
  REAL(fp) :: ke(MAX_N_LAYERS, MAX_N_CLOUDS)  ! Mass extinction coefficient
  REAL(fp) :: w(MAX_N_LAYERS, MAX_N_CLOUDS)   ! Single scatter albedo
  REAL(fp) :: g(MAX_N_LAYERS, MAX_N_CLOUDS)   ! Asymmetry factor
  REAL(fp) :: pcoeff(0:MAX_N_LEGENDRE_TERMS,&
                     MAX_N_PHASE_ELEMENTS,  &
                     MAX_N_LAYERS,          &
                     MAX_N_CLOUDS           ) ! Phase coefficient
  ! The accumulated scattering coefficient
  REAL(fp) :: Total_bs(MAX_N_LAYERS)          ! Volume scattering coefficient
END TYPE CRTM_CSVariables_type
```

Figure 1.3: The structure definition to hold all the forward model CloudScatter results showing the added `csi` structure array used to hold the intermediate interpolation results. The array dimensions are declared in the `CRTM_Parameters` module.

```
TYPE :: ASinterp_type
  ! The interpolating polynomials
  TYPE(LPoly_type) :: wlp  ! Frequency
  TYPE(LPoly_type) :: xlp  ! Effective radius
  ! The LUT interpolation indices
  INTEGER :: i1, i2        ! Frequency
  INTEGER :: j1, j2        ! Effective radius
  ! The LUT interpolation boundary check
  LOGICAL :: f_outbound    ! Frequency
  LOGICAL :: r_outbound    ! Effective radius
  ! The interpolation input
  REAL(fp) :: f_int        ! Frequency
  REAL(fp) :: r_int        ! Effective radius
  ! The data to be interpolated
  REAL(fp) :: f(NPTS)      ! Frequency
  REAL(fp) :: r(NPTS)      ! Effective radius
END TYPE ASinterp_type
```

Figure 1.4: The structure definition to hold the forward model aerosol optical properties lookup table interpolation results.

```
TYPE :: CRTM_ASVariables_type
  PRIVATE
  ! The interpolation data
  TYPE(ASinterp_type) :: asi(MAX_N_LAYERS, MAX_N_AEROSOLS)
  ! The interpolation result
  REAL(fp) :: ke(MAX_N_LAYERS, MAX_N_AEROSOLS)  ! Mass extinction coefficient
  REAL(fp) :: w(MAX_N_LAYERS, MAX_N_AEROSOLS)   ! Single scatter albedo
  REAL(fp) :: g(MAX_N_LAYERS, MAX_N_AEROSOLS)   ! Asymmetry factor
  REAL(fp) :: pcoeff(0:MAX_N_LEGENDRE_TERMS,&
                     MAX_N_PHASE_ELEMENTS,  &
                     MAX_N_LAYERS,          &
                     MAX_N_AEROSOLS         )   ! Phase coefficient
  ! The accumulated scattering coefficient
  REAL(fp) :: Total_bs(MAX_N_LAYERS)            ! Volume scattering coefficient
END TYPE CRTM_ASVariables_type
```

Figure 1.5: The structure definition to hold all the forward model AerosolScatter results showing the added `asi` structure array used to hold the intermediate interpolation results. The array dimensions are declared in the `CRTM_Parameters` module.

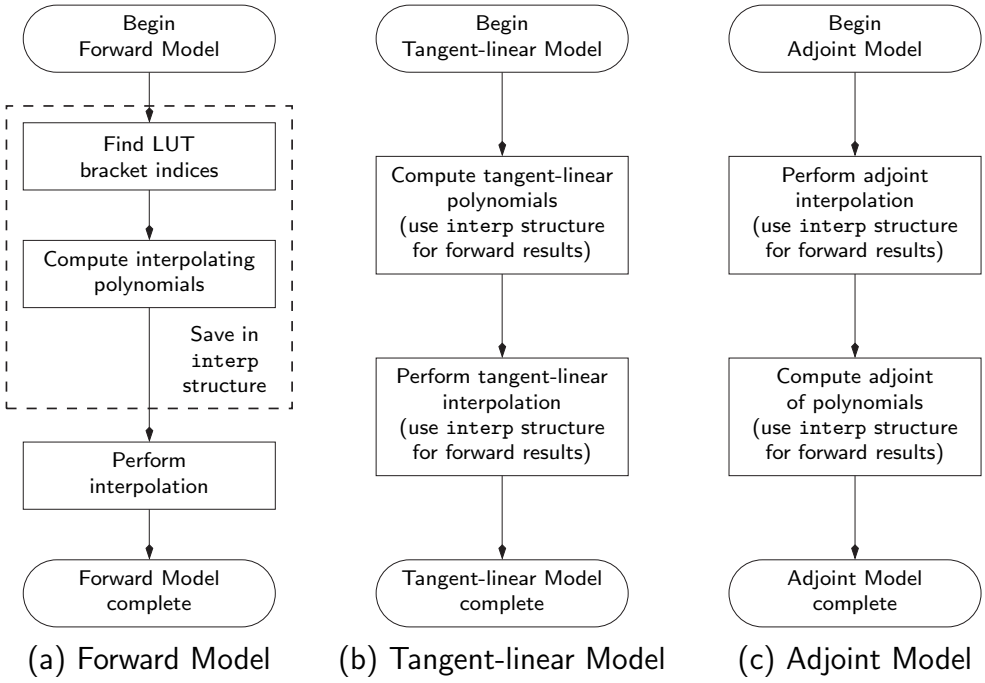

Figure 1.6: Schematic flowcharts of the REL-1.2 CRTM Forward, Tangent-linear, and Adjoint routines used to interpolate the cloud and aerosol optical properties from the CloudCoeff and AerosolCoeff lookup tables. The forward component of the interpolation is now saved in a structure variable and reused in the tangent-linear and adjoint routines.

2 CloudScatter Profile Results

2.1 CloudScatter Forward Model Profile Results

The forward model test results are shown separately for the microwave and infrared cloud optical property LUT interpolation. The microwave interpolation is done at different dimensionalities dependeing on the cloud type. The infrared interpolation is always two-dimensional.

No significant timing differences are seen in the forward model test since the number of calculations is the same, just that some of them are being retained in the `csi` structure as shown in figure 1.3.

	Baseline		Modified	
Routine	**self**	**called**	**self**	**called**
`get_cloud_opt_mw`	76.96	578.58	53.13	571.72
`interp_3d`	26.10	214.69	26.90	210.44
`find_random_index`	136.65	0.00	141.96	0.00
`lpoly`	20.74	96.16	17.99	91.97
`interp_2d`	24.73	56.77	23.12	56.60
`interp_1d`	2.74	0.00	2.74	0.00

Table 2.1: gfortran CloudScatter Forward model profile results for the `Get_Cloud_Opt_MW` subroutine. All times in seconds.

	Baseline		Modified	
Routine	**self**	**called**	**self**	**called**
`get_cloud_opt_ir`	31.69	189.73	23.17	186.95
`interp_2d`	30.76	70.61	28.76	70.40
`find_random_index`	47.62	0.00	49.47	0.00
`lpoly`	7.23	33.51	6.27	32.05

Table 2.2: gfortran CloudScatter Forward model profile results for the `Get_Cloud_Opt_IR` subroutine. All times in seconds.

	Baseline		Modified	
Routine	**self**	**called**	**self**	**called**
`get_cloud_opt_mw`	463.88	4633.15	399.15	4664.71
`interp_3d`	412.34	2133.74	390.48	2148.70
`interp_2d`	383.67	498.79	375.41	512.72
`lpoly`	165.98	598.86	168.51	600.09
`find_random_index`	432.79	0.00	461.62	0.00
`interp_1d`	6.98	0.00	7.18	0.00

Table 2.3: IBM CloudScatter Forward model profile results for the `Get_Cloud_Opt_MW` subroutine. All times in seconds.

2.2 CloudScatter Tangent-Linear Model Profile Results

As with the forward model, the tangent-linear test results are shown separately for the microwave and infrared cloud optical property LUT interpolation.

Here we begin to see significant differences in the routine timings between the Baseline and Modified test runs. Because the forward model interpolation parameters are reused rather than recalculated in the Modified runs,

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_ir	186.83	1514.98	164.67	1533.38
interp_2d	477.22	620.41	466.94	637.73
lpoly	57.84	208.69	58.72	209.12
find_random_index	150.82	0.00	160.87	0.00

Table 2.4: IBM CloudScatter Forward model profile results for the Get_Cloud_Opt_IR subroutine. All times in seconds.

the time required for the find_random_index and lpoly routines do not contribute. Looking at the "self" value for the main routines, we see the Modified times are about 3x faster.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_mw_tl	305.41	1483.90	72.45	1280.26
interp_3d_tl	60.03	754.44	58.53	794.87
interp_2d_tl	53.45	184.56	53.27	199.80
lpoly_tl	34.28	149.79	31.44	142.35
find_random_index	140.22	0.00	-	-
lpoly	19.41	87.73	-	-

Table 2.5: gfortran CloudScatter Tangent-linear model profile results for the Get_Cloud_Opt_MW_TL subroutine. All times in seconds.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_ir_tl	117.27	447.04	27.28	375.34
interp_2d_tl	66.63	230.07	66.26	248.52
lpoly_tl	11.94	52.20	10.96	49.61
find_random_index	48.87	0.00	-	-
lpoly	6.76	30.57	-	-

Table 2.6: gfortran CloudScatter Tangent-linear model profile results for the Get_Cloud_Opt_IR_TL subroutine. All times in seconds.

2.3 CloudScatter Adjoint Model Profile Results

As with the other models, the adjoint test results are shown separately for the microwave and infrared cloud optical property LUT interpolation.

Here again we see significant differences in the routine timings between the Baseline and Modified test runs. Because the forward model interpolation parameters are reused rather than recalculated in the Modified runs, the time required for the find_random_index and lpoly routines do not contribute. Looking at the "self" value for the main routines, we see the Modified times are about 3-4x faster. A word of caution though: the test duration itself is quite short so the speedup factor should be regarded as a very rough guide. The main point is that the modifications do not make the code any slower.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_mw_tl	3886.24	15048.05	859.30	13631.37
interp_3d_tl	1031.52	8794.58	1026.03	8634.92
interp_2d_tl	950.29	1838.77	899.37	1836.93
lpoly_tl	193.76	1038.94	197.15	1036.98
lpoly	164.58	603.85	-	-
find_random_index	431.76	0.00	-	-

Table 2.7: IBM CloudScatter Tangent-linear model profile results for the Get_Cloud_Opt_MW_TL subroutine. All times in seconds.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_ir_tl	1466.88	4324.64	352.51	3833.52
interp_2d_tl	1184.62	2292.19	1118.65	2284.80
lpoly_tl	67.52	362.05	68.70	361.37
lpoly	57.35	210.43	-	-
find_random_index	150.46	0.00	-	-

Table 2.8: IBM CloudScatter Tangent-linear model profile results for the Get_Cloud_Opt_IR_TL subroutine. All times in seconds.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_mw_ad	36.90	187.22	10.22	187.56
interp_3d_ad	8.70	98.79	8.79	120.22
interp_2d_ad	7.82	23.92	8.81	29.42
find_random_index	16.61	0.00	-	-
lpoly_ad	2.72	13.54	2.87	15.18
lpoly	2.25	10.33	-	-

Table 2.9: gfortran CloudScatter Adjoint model profile results for the Get_Cloud_Opt_MW_AD subroutine. All times in seconds.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_ir_ad	16.04	60.53	4.31	59.33
interp_2d_ad	9.74	29.82	10.96	36.60
lpoly_ad	1.66	8.25	1.75	9.24
find_random_index	5.79	0.00	-	-
lpoly	0.79	3.60	-	-

Table 2.10: gfortran CloudScatter Adjoint model profile results for the Get_Cloud_Opt_IR_AD subroutine. All times in seconds.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_mw_ad	509.93	1842.51	101.19	1708.72
interp_3d_ad	126.71	1091.69	119.90	1100.07
interp_2d_ad	112.66	239.15	116.78	239.04
lpoly_ad	13.81	104.49	13.17	106.08
lpoly	19.90	69.37	-	-
find_random_index	51.25	0.00	-	-

Table 2.11: IBM CloudScatter Adjoint model profile results for the Get_Cloud_Opt_MW_AD subroutine. All times in seconds.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_cloud_opt_ir_ad	137.81	564.33	40.58	519.93
interp_2d_ad	140.44	298.13	145.25	297.33
lpoly_ad	8.42	63.68	8.01	64.56
lpoly	6.94	24.17	-	-
find_random_index	17.86	0.00	-	-

Table 2.12: IBM CloudScatter Adjoint model profile results for the Get_Cloud_Opt_IR_AD subroutine. All times in seconds.

3 AerosolScatter Profile Results

3.1 AerosolScatter Forward Model Profile Results

Aerosol optical properties are only used in the infrared spectral region, so only one routine is profiled here – the generic `get_aerosol_opt`. As with the CloudScatter forward model results, no significant timing differences are seen since the number of calculations is essentially the same.

	Baseline		Modified	
Routine	**self**	**called**	**self**	**called**
`get_aerosol_opt`	39.52	230.06	34.28	245.13
`interp_2d`	40.73	91.91	37.85	103.73
`find_random_index`	49.10	0.00	53.28	0.00
`lpoly`	8.01	40.15	9.09	40.75
`aerosol_type_index`	0.18	0.00	0.44	0.00

Table 3.1: gfortran AerosolScatter Forward model profile results for the `Get_Aerosol_Opt` subroutine. All times in seconds.

	Baseline		Modified	
Routine	**self**	**called**	**self**	**called**
`get_aerosol_opt`	267.45	1855.09	234.69	1889.82
`interp_2d`	579.35	817.80	583.05	795.91
`lpoly`	74.27	260.30	73.78	263.76
`find_random_index`	122.63	0.00	172.29	0.00
`aerosol_type_index`	0.74	0.00	1.03	0.00

Table 3.2: IBM AerosolScatter Forward model profile results for the `Get_Aerosol_Opt` subroutine. All times in seconds.

3.2 AerosolScatter Tangent-Linear Model Profile Results

Here we see the decreased execution time of the modified code due to the reuse of the intermediate interpolation results in the tangent-linear model. The speedup is around 3-4x.

	Baseline		Modified	
Routine	**self**	**called**	**self**	**called**
`get_aerosol_opt_tl`	152.97	549.00	41.19	498.31
`interp_2d_tl`	85.45	285.46	98.44	315.26
`lpoly_tl`	15.46	66.40	15.01	69.23
`find_random_index`	51.30	0.00	-	-
`lpoly`	8.45	36.11	-	-
`aerosol_type_index`	0.38	0.00	0.37	0.00

Table 3.3: gfortran AerosolScatter Tangent-linear model profile results for the `Get_Aerosol_Opt_TL` subroutine. All times in seconds.

3.3 AerosolScatter Adjoint Model Profile Results

As with the AerosolScatter tangent-linear model, we see the decreased execution time of the modified code due to the reuse of the intermediate interpolation results in the adjoint model. The speedup is around 2-3x, although

Routine	Baseline self	Baseline called	Modified self	Modified called
get_aerosol_opt_tl	1944.73	5353.09	436.08	4861.03
interp_2d_tl	1380.02	2944.39	1423.29	2896.04
lpoly_tl	85.19	480.71	86.00	454.49
lpoly	73.51	264.22	-	-
find_random_index	124.16	0.00	-	-
aerosol_type_index	0.88	0.00	1.21	0.00

Table 3.4: IBM AerosolScatter Tangent-linear model profile results for the Get_Aerosol_Opt_TL subroutine. All times in seconds.

the total test time is relatively short so care must be taken in interpreting those factors. As with the CloudScatter updates, the main point here is that the modifications do not make the code slower.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_aerosol_opt_ad	28.84	117.95	9.86	92.47
interp_2d_ad	19.48	57.33	19.22	53.72
lpoly_ad	3.26	16.92	3.39	14.75
lpoly	1.80	7.77	-	-
find_random_index	9.38	0.00	-	-
aerosol_type_index	0.05	0.00	0.05	0.00

Table 3.5: gfortran AerosolScatter Adjoint model profile results for the Get_Aerosol_Opt_AD subroutine. All times in seconds.

Routine	Baseline self	Baseline called	Modified self	Modified called
get_aerosol_opt_ad	290.81	1064.11	88.80	985.50
interp_2d_ad	269.14	561.43	276.07	563.02
lpoly_ad	15.95	124.51	15.52	121.79
lpoly	13.22	48.02	-	-
find_random_index	22.79	0.00	-	-
aerosol_type_index	0.19	0.00	0.16	0.00

Table 3.6: IBM AerosolScatter Adjoint model profile results for the Get_Aerosol_Opt_AD subroutine. All times in seconds.

4 CRTM ComponentTest Profile Results

The CRTM ComponentTest test programs run the CRTM models for a variety of sensors (both microwave and infrared) and input profiles (including both cloud and aersol data in the input atmospheric profiles. These tests were run using both the baseline and modified cloud and aerosol optical property interpolation codes to determine what the timing changes would be for a complete model run. Currently only linux/gfortran test cases are available, but the results of the tests performed indicate that the modified interpolation code has no significant impact on the code speed.

The results of the forward/tangent-linear and tangent-linear/adjoint ComponentTest runs are shown in tables 4.1 and 4.2 respectively. In both cases the gas absorption and matrix inversion procedures (the latter used in the CRTM ADA RTSolution computation) were the most expensive procedures. Only those procedures that *did not* call any child processes are shown. For example, the time spent in the `crtm_rtsolution::crtm_doubling_layer` procedure was greater than that of `crtm_utility::matinv` procedure, but the former calls the latter.

4.1 Forward/Tangent-linear ComponentTest Profile Results

	Baseline		Modified	
Routine	**%**	**self time**	**%**	**self time**
`crtm_utility::matinv`	66.7	284849.04	66.0	285413.03
`crtm_atmabsorption::crtm_compute_atmabsorption`	9.0	38424.78	8.9	38527.22
...	...	...	...	...
`crtm_interpolation::interp_1d`	0.9	4009.04	1.0	4356.03
`crtm_interpolation::interp_1d_tl`	0.5	2059.62	0.5	2232.06

Table 4.1: gfortran CRTM Forward/Tangent-linear ComponentTest profile results comparing the most expensive routines with the two most expensive interpolation routines. The routine names are prefixed with their containing module name. All times in seconds.

4.2 Tangent-linear/Adjoint ComponentTest Profile Results

	Baseline		Modified	
Routine	**%**	**self time**	**%**	**self time**
`crtm_atmabsorption::crtm_compute_atmabsorption`	19.5	618.28	19.8	620.95
`crtm_utility::matinv`	11.3	359.16	11.9	372.32
...	...	...	...	...
`crtm_interpolation::interp_1d_tl`	2.1	67.15	2.1	65.79
`crtm_interpolation::interp_1d`	1.8	57.65	2.1	66.17

Table 4.2: gfortran CRTM Tangent-linear/Adjoint ComponentTest profile results comparing the most expensive routines with the two most expensive interpolation routines. The routine names are prefixed with their containing module name. All times in seconds.